Efficient Optimization of the Parameters of LS-SVM for Regression versus Cross-Validation Error

Ginés Rubio[1], Héctor Pomares, Ignacio Rojas, Luis Javier Herrera, and Alberto Guillén

Department of Computer Architecture and Technology, University of Granada, C/ Periodista Daniel Saucedo sn, 18071 Granada, Spain
grubio@atc.ugr.es

Abstract. Least Squares Support Vector Machines (LS-SVM) are the state of the art in kernel methods for regression and function approximation. In the last few years, these models have been successfully applied to time series modelling and prediction. A key issue for the good performance of a LS-SVM model are the values chosen for both the kernel parameters and its hyperparameters in order to avoid overfitting the underlying system to be modelled. In this paper an efficient method for the evaluation of the cross validation error for LS-SVM is revised. The expressions for its partial derivatives are presented in order to improve the procedure for parameter optimization. Some initial guesses to set the values of both kernel parameters and the regularization factor are also presented. We finally conduct some experiments on a time series data example using a number of methods for parameter optimization for LS-SVM models. The results show that the proposed partial derivatives and heuristics can improve the performance with respect to both execution time and the optimized model obtained.

1 Introduction

Least Square Support Vector Machines (LS-SVMs) [1] have been successfully applied to regression and function approximation problems. Although they have proved to be very reliable obtaining good performances and very accurate results, they present some drawbacks:

- The selection of the kernel function could be difficult.
- The optimization of the parameters of the kernel function is computationally intensive.
- The generated models could be huge, because they include all training data inside.

In the literature, the kernel functions used are almost all variants of radial basis functions (RBF) [2] and the analysis of the problem is centered mainly in the feature selection procedure, i.e. which features must be taken into account for

C. Alippi et al. (Eds.): ICANN 2009, Part II, LNCS 5769, pp. 406–415, 2009.

the problem [3] [4]. The values of the parameters in most cases are usually set by optimizing the cross validation error of the models, although some other criteria are available such as the Bayesian Likelihood [5] [6] or some bounds on the training error [7]. In the most popular implementation of LS-SVM for Matlab, LSSVMlab[1], both the Bayesian and the cross validation error-based optimization are available. In the latter case, the software searches in the parameters' space by means of a grid search algorithm, that is very computational intensive for large grid sizes, high cross validation orders and big number of parameters. The exception to this is the special case of the Leave-One-Out cross validation, for which in LSSVMlab an efficient procedure is implemented based on the inversion of the kernel matrix for all the training data. In [8] and [9] expressions for the evaluation of the cross validation error are presented that are efficient and with a computational complexity independent of the order. This paper focus on the efficient and accurate optimization of the cross validation error of arbitrary order versus the parameters using information about the gradient.

The rest of the paper is organized as follows: Section 2 will briefly introduce the LS-SVM model for regression, and we will provide expressions for the evaluation of the cross validation error and for the partial derivatives of the cross validation error with respect to the parameters of the model; Section 3 will tackle the same objectives, but for the special case of unbiased LS-SVM models; Section 4 will present some initial guesses for the non-linear parameters of the model; Finally, in Sections 5 and 6 some experiments will be presented and final conclusions will be drawn.

2 Least Squares Support Vector Machines (LS-SVM)

Given a set of function samples of the form $\{(x_1, y_1), \ldots, (x_n, y_n)\} \subset X \times \mathbb{R}$, where n is the number of samples, the classic linear LS-SVM model for regression relates inputs X with the output Y by minimizing the objective function:

$$\begin{aligned} \min_{w \in X,\ b, e_i \in \mathbb{R}} \tau(w, b, e) &= \tfrac{1}{2}\|w\|^2 + \gamma \tfrac{1}{2} \sum_{i=1}^{n} e_i^2, \\ (y_i - (\langle w, x \rangle + b)) &= e_i, \quad \forall i = 1, \ldots, n. \end{aligned} \tag{1}$$

where w, b are the parameters of the linear approximator, $\gamma > 0$ is a regularization parameter and e_i is the error for the i-th sample ($e = \{e_1, e_2, \cdots, e_n\}$). Since this is a typical problem of optimization of a differentiable function with restrictions, it can be solved by using Lagrange multipliers (α_i), leading to:

$$\left[\begin{array}{c|c} 0 & 1_N^T \\ \hline 1_N & \Omega + I/\gamma \end{array}\right] \left[\begin{array}{c} b \\ \hline \alpha \end{array}\right] = \left[\begin{array}{c} 0 \\ \hline y \end{array}\right] \tag{2}$$

where $\Omega_{ij} = \langle x_i, x_j \rangle$ and I is the identity. Applying the well-known *Kernel Trick*, $\Omega_{ij} = k(x_i, x_j)$, the problem can be extended to non linear cases. The modelled

[1] `http://www.esat.kuleuven.ac.be/sista/lssvmlab/`

function can be written in terms of coefficients α_i, b and values of the kernel function k:

$$f(x) = \sum_{i=1}^{n} \alpha_i k(x, x_i) + b. \tag{3}$$

2.1 Evaluation of the l-fold Cross-Validation Error for LS-SVM

In order to avoid that the LS-SVM model overfits the data it learns from, it is important to measure its performance when approximating data which have not been used during the training process. One of the most common methods of obtaining a model with good generalization capabilities consist in minimizing not just the training error but the cross-validation error. The l-fold cross-validation error of a model is obtaining by dividing the available data in l sub-sets and, alternately using one of the sub-sets as test data and the rest as training data. Therefore, a total of l models are trained and cross-validated.

In [9], a reduced cost method for the evaluation of l-fold cross-validation error for LS-SVM is presented. To compute the cross-validation error of order l the following expression can be used:

$$MSE_{l\text{-fold}} = \frac{1}{n} \sum_{k=1}^{l} \sum_{j=1}^{|\beta^{(m)}|} {\beta_j^{(m)}}^2 \tag{4}$$

where $\beta = [\beta^{(1)}, \beta^{(2)}, \cdots, \beta^{(l)}]^T$ is defined as $\beta^{(m)} = C_{mm}^{-1} \alpha^{(m)}$. Please, note that $\alpha = [\alpha^{(1)}, \alpha^{(2)}, \cdots, \alpha^{(l)}]^T$ here is not the same of equation (2), but given from:

$$\alpha = Cy \tag{5}$$

$$C = \begin{bmatrix} C_{11} & C_{12} & \cdots & C_{1l} \\ C_{12}^T & C_{22} & \cdots & C_{2l} \\ \vdots & \vdots & \ddots & \vdots \\ C_{l2}^T & C_{2l}^T & \cdots & C_{ll} \end{bmatrix} = K_\gamma^{-1} + \frac{1}{d} K_\gamma^{-1} 1_n 1_n^T K_\gamma^{-1} \tag{6}$$

$$d = -1_n^T K_\gamma^{-1} 1_n \tag{7}$$

$$K_\gamma = K + I/\gamma \tag{8}$$

$$K_{ij} = k(x_i, x_j; \Theta) \tag{9}$$

where I is the identity matrix and Θ is the vector of parameters for the kernel k. The computational cost is independent from l, the fold order, being dominated by the costs of the inversion of the matrix K_γ.

2.2 Partial Derivatives of the l-Fold Cross-Validation Error for LS-SVM

In the last sub-section we have stated that our main objective when optimizing a LS-SVM model with good generalization properties is to minimize the

cross-validation error. In order to accomplish the actual minimization, we need the expressions of the partial derivatives of the l-fold cross-validation error (Eq. 4) with respect to a given parameter p (that can be the regularization factor γ or any kernel parameter). After some algebra, we obtain the following expressions:

$$\frac{\partial MSE_{l\text{-fold}}}{\partial p} = \frac{2}{n}\sum_{k=1}^{l}\sum_{j=1}^{|\beta^{(m)}|}\beta_j^{(m)}\left[\frac{\partial \beta^{(m)}}{\partial p}\right]_j \tag{10}$$

$$\frac{\partial \beta^{(m)}}{\partial p} = \frac{\partial C_{mm}^{-1}}{\partial p}\alpha^{(m)} + C_{mm}^{-1}\frac{\partial \alpha^{(m)}}{\partial p} \tag{11}$$

$$\frac{\partial \alpha^{(m)}}{\partial p} = \frac{\partial C_{mm}}{\partial p}y^{(m)} \tag{12}$$

$$\frac{\partial C_{mm}^{-1}}{\partial p} = -C_{mm}^{-1}\frac{\partial C_{mm}}{\partial p}C_{mm}^{-1} \tag{13}$$

$$\frac{\partial C}{\partial p} = \begin{bmatrix} \frac{\partial C_{11}}{\partial p} & \frac{\partial C_{12}}{\partial p} & \cdots & \frac{\partial C_{1l}}{\partial p} \\ \frac{\partial C_{12}^T}{\partial p} & \frac{\partial C_{22}}{\partial p} & \cdots & \frac{\partial C_{2l}}{\partial p} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial C_{l2}^T}{\partial p} & \frac{\partial C_{2l}^T}{\partial p} & \cdots & \frac{\partial C_{ll}}{\partial p} \end{bmatrix} \tag{14}$$

$$\frac{\partial C}{\partial p} = \frac{\partial K_\gamma^{-1}}{\partial p} + \frac{\partial d^{-1}}{\partial p}K_\gamma^{-1}1_n1_n^TK_\gamma^{-1}\cdots \tag{15}$$

$$+\frac{1}{d}\left(\frac{\partial K_\gamma^{-1}}{\partial p}1_n1_n^TK_\gamma^{-1} + K_\gamma^{-1}1_n1_n^T\frac{\partial K_\gamma^{-1}}{\partial p}\right) \tag{16}$$

$$\frac{\partial d^{-1}}{\partial p} = \frac{1}{d^2}1_n^T\frac{\partial K_\gamma^{-1}}{\partial p}1_n \tag{17}$$

$$\frac{\partial K_\gamma^{-1}}{\partial p} = -K_\gamma^{-1}\frac{\partial K_\gamma}{\partial p}K_\gamma^{-1} \tag{18}$$

Therefore, as expected, the partial derivative of the cross-validation error with respect to parameter p depends on $\frac{\partial K_\gamma}{\partial p}$, the partial derivative of $K_\gamma = K + I/\gamma$. For each $p = \theta \in \Theta$, the derivative depends on the specific kernel function k, but for $p = \gamma$ it is $\frac{\partial K_\gamma}{\partial p} = -I/\gamma^2$.

The evaluation of the partial derivatives implies the inversion of the matrix K_γ, that has a computational complexity of $O(N) = N^3$ with exact methods, and depends on the computation of the partial derivatives of the kernel function with respect to its parameters, that must be provided for each particular kernel.

In sum, with the use of the above equations a Conjugate Gradient (CG) scheme can be used to optimize both the regularization factor γ and the kernel parameters in a LS-SVM model. We should here point out that in [1] it is recommended to optimize in different levels the kernel parameters and the regularization factor γ). As we are not guaranteed the convergence to the global optimum, it is recommended to use several optimization stages from different

starting points. In section 4 we will also provide some initial guesses for these starting points in order to provide a faster route to the global solution.

3 Unbiased LS-SVM

For function approximation, it is often preferred to use an unbiased version of LS-SVM. The elimination of the bias simplifies the expressions, considering that the functions to model have zero mean, a condition that can be imposed without loss of generality. Given a set of function samples $\{(x_1, y_1), \ldots, (x_n, y_n)\} \subset X \times \mathbb{R}$, where n is the number of samples, the unbiased LS-SVM model for regression relates inputs X with the output Y by the following optimization problem:

$$\begin{aligned} \min_{w,e_i \in \mathbb{R}} \tau(w,e) &= \tfrac{1}{2}\|w\|^2 + \gamma \tfrac{1}{2} \sum_{i=1}^{n} e_i^2, \\ (y_i - (\langle w, x \rangle)) &= e_i, \quad \forall i = 1, \ldots, n. \end{aligned} \tag{19}$$

that leads again to the use of Lagrange multipliers α and, finally, to solve the linear system:

$$[\Omega + I/\gamma]\,[\alpha] = [y] \tag{20}$$

where $\Omega_{ij} =< x_i, x_j >$, the scalar product between a pair of input vector, I is the identity matrix and γ is the regularization factor. If the scalar product operation in the *input space* is substituted by a scalar product in a *feature space* given by a kernel function $k(x, x') =< \phi(x), \phi(x') >$, where ϕ is the function that map points from input space to the feature space, then $\Omega_{ij} = k(x_i, x_j)$ and the actual model would be given by:

$$f(x) = \sum_{i=1}^{n} \alpha_i k(x, x_i). \tag{21}$$

3.1 *l*-Fold Cross-Validation Error for Unbiased LS-SVM

The reduced cost method to evaluate the l-fold cross-validation error in the special case of unbiased LS-SVM is:

$$MSE_{l\text{-fold}} = \frac{1}{n} \sum_{k=1}^{l} \sum_{j=1}^{|\beta^{(m)}|} {\beta_j^{(m)}}^2 \tag{22}$$

$$\beta^{(m)} = C_{mm}^{-1} \alpha^{(m)} \tag{23}$$

$$C = K_\gamma^{-1} \tag{24}$$

where α is the same of equation (20) and K_γ the same of equation 8.

3.2 Partial Derivatives of the l-Fold Cross-Validation Error for Unbiased LS-SVM

Again, in the special case of using unbiased LS-SVM, the partial derivatives of the l-fold cross-validation error (22) with respect to a given parameter p (that can be γ or a kernel parameter) would stay as:

$$\frac{\partial MSE_{l\text{-fold}}}{\partial p} = \frac{2}{n}\sum_{k=1}^{l}\sum_{j=1}^{|\beta^{(m)}|}\beta_j^{(m)}\left[\frac{\partial \beta^{(m)}}{\partial p}\right]_j \tag{25}$$

$$\frac{\partial \beta^{(m)}}{\partial p} = \frac{\partial C_{mm}^{-1}}{\partial p}\alpha^{(m)} + C_{mm}^{-1}\frac{\partial \alpha^{(m)}}{\partial p} \tag{26}$$

$$\frac{\partial \alpha}{\partial p} = \frac{\partial K_\gamma^{-1}}{\partial p}y \tag{27}$$

4 Guesses for the Initialization of the LS-SVM Parameters

In this section, some guesses are given to initialize the regularization factor γ from training data. Nevertheless, it is not possible to do the same for the parameters of any arbitrary kernel. Therefore, we will concentrate on the most common case, the Radial Basis Function (RBF) kernel (29) which has only one parameter, σ.

4.1 Initialization of the Regularization Parameter γ

In the paper [6] it is shown that the optimization problem of a LS-SVM model given by equation (19) can be rewritten as:

$$\begin{aligned}\min_{w,e_i\in\mathbb{R}} \tau(w,e) &= \mu\tfrac{1}{2}\|w\|^2 + \xi\tfrac{1}{2}\sum_{i=1}^{n}e_i^2,\\ (y_i - (\langle w,\phi(x)\rangle)) &= e_i, \quad \forall i = 1,\ldots,n.\end{aligned} \tag{28}$$

where $\gamma = \xi/\mu$, $1/\xi = \sigma_e^2$, where σ_e^2 is the variance of the noise in the data and μ is the parameter that controls the regularization of the model.

Without prior information to set the μ value, we let μ be set to 1 provided all data is normalized with mean 0 and standard deviation 1. The ξ parameter depends on the variance of the noise in the data. We can estimate this noise variance, $\hat{\sigma}_e^2$, using the non parametric estimators delta or gamma test, as recommended in [10] in which it is also pointed out that the gamma test is more reliable than the delta test for data of dimensionality higher than four. Thus, our initial guess for the regularization factor γ in a LS-SVM model will be $\gamma_h = 1/\hat{\sigma}_e^2$, where $\hat{\sigma}_e^2$ is evaluated from the data using the delta or the gamma test.

In the worst case, all data are pure noise, so $\sigma_e^2 = \sigma_y^2$ and a minimum γ value can be set as $\gamma_{min} = 1/\sigma_y^2$. A reasonable range to search for an initial value of γ

could be $[\gamma_{min}, \gamma_h]$, but if we rely on non parametric noise estimation methods we can use values very close to γ_h.

In [11] it is demonstrated that the presence of noise in the input vectors makes that the delta and gamma test, instead of approximating the variance of the noise, give a measure of the variance of the effective noise in the output. The value of the variance of the effective noise is larger than the variance of the noise itself, but a better approximation cannot be performed anyway.

Since in order to use regression methods for time series prediction the input/ouput vectors have to be generated from the time series, the input vectors will always have noise. So a reasonable range to search the γ value in a range centered on γ_h could be $[0.5 \cdot \gamma_h, 2 \cdot \gamma_h]$.

4.2 Initial Guess for the σ Parameter of the RBF Kernel

Considering the most common case of LS-SVM model for regression, i.e., the LS-SVM model with RBF kernel, whose equation is:

$$k(x, x') = \exp\left(-\frac{1}{\sigma^2}\|x - x'\|^2\right). \tag{29}$$

the σ value is related to the distance between training points and to the smooth interpolation of the resulting model from the I/O data: higher values of σ give smoother functions. It can be expected to find a good σ in $[\sigma_{min}, \sigma_{max}]$, where σ_{min} is the minimum distance (non zero) between 2 training points and σ_{max} is the maximum distance between 2 training points. But a smaller, but reasonable, parameter search space for σ could be the range $[0.5 \cdot \overline{\sigma}, 2 \cdot \overline{\sigma}]$, centered in $\overline{\sigma} = 0.5 \cdot (\sigma_{max} + \sigma_{min})$ that can be used as a good enough starting point.

In order to use this kernel in the expressions obtained in the previous sections, we need the derivative of the kernel function with respect to its parameter σ:

$$\frac{\partial k(x, x')}{\partial \sigma} = \exp\left(-\frac{1}{\sigma^2}\|x - x'\|^2\right)\frac{1}{\sigma^4}\|x - x'\|^2. \tag{30}$$

5 Experiments

To test the proposed optimization methods and the heuristics for the initialization of the parameters for LS-SVM models with RBF kernel, the Mackey-Glass time series is used. The Mackey-Glass time series [12] is a very well-known benchmark for time series modelling and prediction. It is an artificial time series without noise, that has been widely used in the literature for the comparison of neuro-fuzzy and other nonlinear models. In our experiments, 1000 data points of the series, $y_1, ..., y_{1000}$, were taken, and normalized with zero mean and unit variance. Gaussian additive noise of zero mean and controlled variance $\sigma_e^2 = \{0, 1, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}\}$ was added to this sequence of data, $\hat{y}_i = y_i + N(0, \sigma_e^2)$, in order to test the non parametric noise estimation, and confirm that the presence of noise in the inputs can make it fail. From

Table 1. Optimizations with RBF kernel using heuristics for Mackey-Glass with noise on all series. σ_e^2: variance of the added Gaussian noise; NNE: evaluated variance of noise with non parametric noise estimation; $NRMSE_{NNE}$ best training error that can be reached evaluated from NNE; Method: Heuristic, using $\gamma_h, \overline{\sigma}$, Gridsearch using reduced cost evaluation of 10 fold cross-validation error in range $[0.5 \cdot \gamma_h, 2 \cdot \gamma_h]$, BFGS applied from $\gamma_h, \overline{\sigma}$, Iterative Local Search (of BFGS) applied from $\gamma_h, \overline{\sigma}$; T: time in seconds; f.e.: number of cross-validation error evaluations performed; 10-CVE: final value of 10 fold cross-validation error reached. Best values of T and 10-CVE in bold.

		Method	Training	Test	T	f.e.	10-CVE
		Heuristic	5.58e-03	4.95e-03	2.10e-01	0	1.38e-04
σ_e^2	0	GS	5.51e-03	4.88e-03	1.33e+01	100	1.37e-04
NNE	1.00e-06	BFGS	4.49e-03	4.03e-03	**6.99e+00**	22	1.25e-04
$NRMSE_{NNE}$	1.00e-03	ILS	4.49e-03	4.03e-03	4.47e+01	100	**1.25e-04**
		LSSVMlab	5.61e-03	4.97e-03	3.61e+02	100	1.38e-04
		Heuristic	1.19e+00	1.27e+00	1.91e-01	0	1.53e+00
σ_e^2	1	GS	1.20e+00	1.27e+00	1.34e+01	100	1.52e+00
NNE	1.38e+00	BFGS	1.20e+00	1.27e+00	**3.68e+00**	12	**1.52e+00**
$NRMSE_{NNE}$	1.17e+00	ILS	1.20e+00	1.27e+00	3.22e+01	100	1.52e+00
		LSSVMlab	1.20e+00	1.27e+00	3.55e+02	100	1.53e+00
		Heuristic	4.16e-01	4.27e-01	1.91e-01	0	2.02e-01
σ_e^2	1.00e-01	GS	4.23e-01	4.29e-01	1.34e+01	100	2.01e-01
NNE	1.86e-01	BFGS	4.21e-01	4.27e-01	**3.40e+00**	11	**2.00e-01**
$NRMSE_{NNE}$	4.31e-01	ILS	4.21e-01	4.27e-01	3.10e+01	100	2.00e-01
		LSSVMlab	4.18e-01	4.28e-01	3.56e+02	100	2.02e-01
		Heuristic	1.45e-01	1.46e-01	1.91e-01	0	2.55e-02
σ_e^2	1.00e-02	GS	1.31e-01	1.48e-01	1.34e+01	100	2.45e-02
NNE	1.63e-02	BFGS	1.33e-01	1.46e-01	**4.33e+00**	14	**2.44e-02**
$NRMSE_{NNE}$	1.28e-01	ILS	1.33e-01	1.46e-01	3.10e+01	100	2.44e-02
		LSSVMlab	1.45e-01	1.46e-01	3.61e+02	100	2.55e-02
		Heuristic	5.38e-02	5.60e-02	1.91e-01	0	3.74e-03
σ_e^2	1.00e-03	GS	4.72e-02	5.50e-02	1.34e+01	100	3.42e-03
NNE	2.44e-03	BFGS	4.79e-02	5.55e-02	**5.55e+00**	18	**3.41e-03**
$NRMSE_{NNE}$	4.94e-02	ILS	4.79e-02	5.55e-02	2.91e+01	100	3.41e-03
		LSSVMlab	5.39e-02	5.61e-02	3.62e+02	100	3.74e-03
		Heuristic	1.77e-02	2.19e-02	1.91e-01	0	5.43e-04
σ_e^2	1.00e-04	GS	1.64e-02	2.09e-02	1.35e+01	100	5.27e-04
NNE	3.18e-05	BFGS	1.61e-02	2.07e-02	**4.34e+00**	14	**5.26e-04**
$NRMSE_{NNE}$	5.64e-03	ILS	1.61e-02	2.07e-02	2.80e+01	100	5.26e-04
		LSSVMlab	1.77e-02	2.19e-02	3.62e+02	100	5.43e-04

the noisy data $\hat{y}_1, ..., \hat{y}_{1000}$, inputs and output vectors were created in the form $[\hat{y}_{t-24}, \hat{y}_{t-18}, \hat{y}_{t-12}, \hat{y}_{t-6}, \hat{y}_t]$, and half of the vectors were used for training and half for test. For the LS-SVM models, the heuristics were applied and tested using the reduced-cost evaluation procedure of the 10 fold cross-validation error (which is one of the most common orders used in literature):

1. Directly: fixing the values of the parameters $\gamma_h, \overline{\sigma}$.
2. A grid-search procedure: the same used on LSSVMlab with the range suggested in Section 4.
3. A local search procedure: using the Broyden-Fletcher-Goldfarb-Shanno method, BFGS, with initial point $(\gamma_h, \overline{\sigma})$
4. A global search procedure: Iterative Local Search using as local search BFGS with initial point $(\gamma_h, \overline{\sigma})$.

The LSSVMlab toolbox was also used. The training and test errors are given independently from translation and scale factors using the Normalized Root Mean Square Error [13] ($NRMSE = \sqrt{MSE/\sigma_y^2}$, where MSE is the Mean Square Error and σ_y^2 the variance of the function to model). The evaluated best training NRMSE error computed from the non parametric noise error will also be shown. All the software was implemented in Matlab, and the experiments were executed on a personal computer with an Intel Core 2 Quad CPU at 2.83GHz and 8 GB of RAM. Finally, the number of evaluations of the cross-validation procedure was limited to 100 in the experiments.

The results obtained are shown in Table 1. As can be seen from the table, the procedure to evaluate the cross-validation error from [9] is faster than the naive implementation for this case (see rows labelled *GS* y *LSSVMlab*). It is also shown that the evaluation of the partial derivatives (used in *BFGS* and *ILS*) adds some overhead to the evaluation of the cross-validation error but allows us to reach a local optima with less number of evaluations. The initial point for the optimization procedures has an important influence on the results. In this case the global search procedure (*ILS*) only reached a better solution than a local search procedure (*BFGS*) in the case of noise-free data. It is worth to recall that the number of evaluations required by BFGS to converge is quite smaller than the maximum allowed in most cases (exceptions for $\sigma_e^2 = 1.0e-2$ y $\sigma_e^2 = 1.0e-4$). It is remarkable the relative good results reached using only the heuristic initialization of the parameters in the models (rows labelled *Heuristic*).

6 Conclusions

In this paper we have obtained the expressions of the derivatives of the reduced cost evaluation of an arbitrary order cross-validation error for LS-SVM models. Some guesses have also been given for the initial values of the regularization factor γ of the LS-SVM model and for the RBF kernel parameter σ. All the expressions were implemented in Matlab. In the experiments we have compared the results obtained by a derivative-free procedure (gridsearch), a conjugate gradient procedure, and a global search procedure (Simulated Annealing guided by gradient). Time and error measures were taken and compared with the most popular implementation of LS-SVM for matlab, LS-SVMlab, for the well-known Mackey-Glass time series. The results confirm that the use of our reduced-cost expressions achieves performances that outperforms the naive implementation

used in LS-SVMlab, and that gradient-based procedures can optimize the parameters more efficiently that other methods that don't use this information. The validity of the given heuristics has also been confirmed for this example.

Acknowledgment

This work has been supported by the Spanish CICYT Project TIN2007-60587 and Junta Andalucia Project P07-TIC-02768.

References

1. Suykens, J.A.K., Van Gestel, T., De Brabanter, J., De Moor, B., Vandewalle, J.: Least Squares Support Vector Machines. World Scientific, Singapore (2002)
2. Rojas, I., et al.: Analysis of the functional block involved in the design of radial basis function networks. Neural Processing Letters 12, 1–17 (2000)
3. Müller, K.-R., Smola, A.J., Rätsch, G., Schölkopf, B., Kohlmorgen, J., Vapnik, V.: Using support vector machines for time series prediction (2000)
4. Rubio, G., Guillen, A., Herrera, L.J., Pomares, H., Rojas, I.: Use of specific-to-problem kernel functions for time series modeling. In: ESTSP 2008: Proceedings of the European Symposium on Time Series Prediction, pp. 177–186 (2008)
5. Scholkopf, B., Smola, A.J.: Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. MIT Press, Cambridge (2001)
6. Van Gestel, T., Suykens, J.A.K., Baestaens, D.-E., Lambrechts, A., Lanckriet, G., Vandaele, B., De Moor, B., Vandewalle, J.: Financial time series prediction using least squares support vector machines within the evidence framework. IEEE Transactions on Neural Networks 12(4), 809–821 (2001)
7. Lendasse, A., Ji, Y., Reyhani, N., Verleysen, M.: Ls-svm hyperparameter selection with a nonparametric noise estimator. In: ICANN (2), pp. 625–630 (2005)
8. Ying, Z., Keong, K.C.: Fast leave-one-out evaluation and improvement on inference for ls-svms. In: Proceedings of the 17th International Conference on Pattern Recognition, ICPR 2004, vol. 3, pp. 494–497 (2004)
9. An, S., Liu, W., Venkatesh, S.: Fast cross-validation algorithms for least squares support vector machine and kernel ridge regression. Pattern Recogn. 40(8), 2154–2162 (2007)
10. Liitiäinen, E., Lendasse, A., Corona, F.: Non-parametric residual variance estimation in supervised learning. In: Sandoval, F., Prieto, A.G., Cabestany, J., Graña, M. (eds.) IWANN 2007. LNCS, vol. 4507, pp. 63–71. Springer, Heidelberg (2007)
11. Jones, A.J., Evans, D., Kemp, S.E.: A note on the Gamma test analysis of noisy input/output data and noisy time series. Physica D Nonlinear Phenomena 229, 1–8 (2007)
12. Mackey, M.C., Glass, L.: Oscillation and Chaos in Physiological Control Systems. Science 197(4300), 287–289 (1977)
13. Herrera, L.J., et al.: TaSe, a Taylor Series-based fuzzy system model that combines interpretability and accuracy. Fuzzy Sets and Systems 153, 403–427 (2005)